```
function est=imp_sampling(param, param_prop, data, D, M, ub, lb, seed, options)
    consumer = data(:,1);
    N_cons = length(unique(consumer));
    N_prod = data(:,end-2);

    Js = unique(N_prod);
    Num_J = length(Js);
    table=importdata('tableZ.csv');%comment unless using look-up table method
    W_PROP = zeros(N_cons, M, Num_J);
    L_ALL = zeros(N_cons, M, Num_J);
    C = {};
    WEIGHT = {};

    % Proposal paramters
    theta_mean1 = param_prop(1, 1:4);
    theta_std1 = param_prop(1, 6:9);
    c_mean1 = param_prop(1, 5);
    c_std1 = param_prop(1, 10);

    %construct likelihood for consumers with the same number of searches
    for i = 1:Num_J
        nalt = Js(i);
        dat = data(N_prod == nalt,:);
        N_obs=length(dat);

        % Choices
        tran = dat(:, end);
        searched = dat(:, end - 1);
        last = dat(:, end - 4);
        has_searched = dat(:, end - 3);
        outside = dat(:, 3);
        X = dat(:,4:7);

        % Simulate Proposal draws
        theta_all = zeros(N_obs, M);
        c_all = zeros(N_obs, M);
        theta_weight = zeros(N_cons*(nalt - 1), M);
        for pd = 1:M
            rng(pd*seed, 'twister');
            b = theta_mean1 + theta_std1.*randn(N_cons, nalt - 1);
            theta_draws = X.*repelem(b, nalt, 1);
            c_draws = exp(c_mean1 + c_std1*randn(N_obs, 1));
            theta_all(:, pd) = sum(theta_draws,2);
            theta_weight(:, pd) = reshape(b', N_cons*(nalt - 1), 1);
            c_all(:, pd) = c_draws;
        end
        C{i} = c_all;
        WEIGHT{i} = theta_weight;

        % Compute poposal weights
```

```
for pd = 1:M
    theta_pd = reshape(theta_weight(:, pd), nalt - 1, N_cons);
    for cons = 1:N_cons
        theta_n_pd = theta_pd(:, cons)';
        c_n_pd = c_all(consumer == cons, pd);

        w_theta = normpdf(theta_n_pd, theta_mean1, theta_std1);
        w_c = normpdf(log(c_n_pd), c_mean1, c_std1);
        W_PROP(cons, pd, i) = prod(w_theta)*prod(w_c);
    end
end

% Compute likelihood at proposal draws
for pd = 1:M
    rng(seed*pd);
    epsilonDraw=randn(N_obs,D); %D draws for every M
    etaDraw=randn(N_obs,D);
    c = c_all(:, pd);
    eut = (repmat(theta_all(:, pd), 1, D) + etaDraw).*(1 - outside);
    ut = eut + epsilonDraw;

    %%%%%%FORM Z's%%%%%%%%%
    %%%1. look-up table method
    m=zeros(N_obs,1);
    for j=1:N_obs
        lookupvalue=abs(table(:,2)-c(j));
        if (table(1,2)>=c(j)&& c(j)>=table(end,2))
            [~,index_m]=min(lookupvalue);
            m(j)=table(index_m,1);
        elseif table(1,2)<c(j)
            m(j)=-c(j);
        elseif c(j)<table(end,2)
            m(j)=4.001;
        end
    end

    %%%%2. newton method
    % m=zeros(N_obs,1);
    % x0 = 0; % initial point
    % for j = 1:size(c, 1)
    %     m(j) = newtonZ(c(j), x0);
    % end

    %%%%3. contraction mapping method
    % m=zeros(N_obs,1); % initial point
    % for i = 1:size(c, 1)
    %     m(i) = contractionZ(m(i), c(i));
    % end

    z = eut + m;
```

```
            %%%%%%%%%%%%%%%%%%%%%
            ut_searched=ut;
            searched2=repmat(searched,1,D);
            ut_searched(searched2==0)=NaN;
            for d = 1:D
                %best ut_so_far
                ymax=cummax(reshape(ut_searched(:,d),nalt,N_cons));
                ymax=circshift(ymax,1);
                ymax=reshape(ymax,N_obs,1);
                %best z_next
                zmax=cummax(reshape(z(:,d),nalt,N_cons),'reverse');
                zmax=circshift(zmax,-1);
                zmax=reshape(zmax,N_obs,1);
                %outside option for each consumer
                u0_2=ut(:,d).*outside;
                u0_3=reshape(u0_2,nalt,N_cons);
                u0_4=repmat(sum(u0_3),nalt,1);
                u0_5=reshape(u0_4,N_obs,1);

                supp_var = ones(size(dat,1),1);
                %selection rule: z>z_next
                order=(z(:,d)-zmax).*has_searched.*searched.*(1-outside).*(1-last)
+ supp_var.*last + supp_var.*outside + supp_var.*(1-has_searched) +
supp_var.*(1-searched);

                order = (order > 0);

                %stopping rule: z>u_so_far
                search_1 = (z(:,d)-ymax).*has_searched.*searched.*(1-outside) +
supp_var.*outside + supp_var.*(1-searched) + supp_var.*(1-has_searched);

                search_2 = (ymax-z(:,d)).*has_searched.*(1-searched) +
supp_var.*(1-has_searched) + supp_var.*searched;

                search_3 = (u0_5-z(:,d)).*(1-has_searched).*(1-outside) +
supp_var.*has_searched + supp_var.*outside;


                search_1 = (search_1 > 0);
                search_2 = (search_2 > 0);
                search_3 = (search_3 > 0);

                %choice rule
                u_ch2=ut(:,d).*tran;
                u_ch3=reshape(u_ch2,nalt,N_cons);
                u_ch4=repmat(sum(u_ch3),nalt,1);
                u_ch5=reshape(u_ch4,N_obs,1);

                choice = (u_ch5-ut(:,d)).*(1-tran).*searched + supp_var.*tran +
supp_var.*(1 - searched);
```

```
                choice = (choice > 0);

                %1. combine all inputs
                chain_mult = order.*search_1.*search_2.*search_2.*search_3.*choice;

                %2. sum/prod at the consumer level
                prob(:,d)  = accumarray(consumer, chain_mult, [N_cons 1],@prod);
            end
            %3. avg across draws (D)
            lk=mean(prob,2);
            L_ALL(:, pd, i) = lk;
        end
    end

    %Maximization
    f = @(x)liklWeitz_imp(x, W_PROP, L_ALL, WEIGHT, C, consumer, Js, D, M, seed);
    %problem has no linear constraints, so set those arguments to []
    A = [];
    b = [];
    Aeq = [];
    beq = [];
    nonlcon = [];
    [be,val,exitflag,output,lambda,grad,hessian] =
fmincon(f,param,A,b,Aeq,beq,lb,ub,nonlcon,options);

    se = real(sqrt(diag(inv(hessian))));
    est = [be'; se; val; exitflag];
end
```